Sí

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/705,678	11/02/2000	Darrell D. Boggs	042390.P9577	6275

7590 07/22/2004

Eric S Hyman
Blakely Sokoloff Taylor & Zafman LLP
12400 Wilshire Boulevard 7th Floor
Los Angeles, CA 90025

EXAMINER	
HUISMAN, DAVID J	
ART UNIT	**PAPER NUMBER**
2183	

DATE MAILED: 07/22/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)
	09/705,678	BOGGS ET AL.
	Examiner	Art Unit
	David J. Huisman	2183

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *21 June 2004*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-19* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-19* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☒ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *24 February 2004* is/are: a)☐ accepted or b)☒ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1-19 have been examined.

Papers Submitted

2. It is hereby acknowledged that the following papers have been received and placed of record in the file: RCE and Extension of Time as received on 6/21/2004.

Specification

3. The title of the invention is not descriptive. A new title is required that is clearly indicative of the invention to which the claims are directed. The examiner recommends incorporating the idea of the counter into the title.

Drawings

4. The drawings are objected to as failing to comply with 37 CFR 1.84(p)(5) because they do not include the following reference character(s) mentioned in the description: On page 7, line 25, of the specification, the applicant lists reference numbers 1011 and 1001. However, these numbers have not been located within the drawings. Corrected drawing sheets are required in reply to the Office action to avoid abandonment of the application. Any amended replacement drawing sheet should include all of the figures appearing on the immediate prior version of the sheet, even if only one figure is being amended. The replacement sheet(s) should be labeled "Replacement Sheet" in the page header (as per 37 CFR 1.84(c)) so as not to obstruct any portion of the drawing figures. If the changes are not accepted by the examiner, the applicant will be

notified and informed of any required corrective action in the next Office action. The objection to the drawings will not be held in abeyance.

Claim Objections

5. Claim 1 is objected to because of the following informalities: Please remove the space in line 12 after "replayed," and before "wherein". Appropriate correction is required.

6. Claim 11 is objected to because of the following informalities: Please remove the space in line 11 after "successfully," and before "wherein". Appropriate correction is required.

Claim Rejections - 35 USC § 112

7. The following is a quotation of the first paragraph of 35 U.S.C. 112:

> The specification shall contain a written description of the invention, and of the manner and process of making and using it, in such full, clear, concise, and exact terms as to enable any person skilled in the art to which it pertains, or with which it is most nearly connected, to make and use the same and shall set forth the best mode contemplated by the inventor of carrying out his invention.

8. Claim 1 is rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the enablement requirement. The claim(s) contains subject matter which was not described in the specification in such a way as to enable one skilled in the art to which it pertains, or with which it is most nearly connected, to make and/or use the invention. More specifically, in claim 1, a counter is associated with an instruction (lines 11-12). However, from lines 13-14, when the counter is less than a predetermined value, instead of executing the associated instruction, which should be the case according to applicant's specification, applicant claims executing different instructions (independent and associated dependent instructions). The examiner recommends

inserting --independent-- before "instruction" in line 11, replacing "wherein independent instructions" with --wherein the independent instruction-- in line 13, and inserting --independent-- before "instruction" in lines 15 and 16.

9. Claim 11 is rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the enablement requirement. The claim(s) contains subject matter which was not described in the specification in such a way as to enable one skilled in the art to which it pertains, or with which it is most nearly connected, to make and/or use the invention. More specifically, in claim 11, a counter is associated with an instruction (lines 7-8). However, from lines 12-13, when the counter is less than a predetermined value, instead of executing the associated instruction, which should be the case according to applicant's specification, applicant claims executing different instructions (independent and associated dependent instructions). The examiner recommends inserting --independent-- before "instruction" in line 7, replacing "wherein independent instructions" with --wherein the independent instruction-- in line 12, and inserting --independent-- before "instruction" in lines 14 and 15.

Claim Rejections - 35 USC § 103

10. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

11. Claims 1-4, 6-10, and 17-18 are rejected under 35 U.S.C. 103(a) as being unpatentable over Sager, U.S. Patent No. 5,966,544 (as applied in the previous Office Action) in view of

Heath et al., U.S. Patent No. 3,603,934 (herein referred to as Heath), and further in view of Hennessy and Patterson, Computer Architecture - A Quantitative Approach, 2nd Edition, 1996 (herein referred to as Hennessy). In addition Hennessy is used to show a teaching of how reservation stations are used in scheduling instructions for execution.

12. Referring to claim 1, Sager has taught a processor comprising:

a) a replay queue to receive a plurality of instructions. See the buffer in Fig.7 and column 9, line 50, to column 10, line 2.

b) an execution unit to execute the plurality of instructions. See Fig.7 and column 8, lines 64-67.

c) a scheduler coupled between the replay queue and the execution unit to speculatively schedule instructions for execution. See Fig.7 and column 10, lines 7-32, and note that the combination of the scheduler and mux selects one of multiple instructions to send to the execution core. It can be seen that the output instruction of the buffer (replay queue) goes to the mux portion of the scheduler, wherein the mux will eventually send that instruction to the execution unit. In addition, although Sager has not explicitly stated that the instructions are scheduled based on data dependencies and expected latencies of said plurality of instructions, Sager has taught that the scheduler may include a reservation station. See column 8, lines 52-63. As is known in the art, and as supported by Hennessy, a reservation station schedules instructions based on latencies and dependencies. For example, regarding latencies, see page 255-256 of Hennessy, and note that the reservation station tracks which execution units are busy. Clearly, if a unit is busy, it will not be able to execute a next instruction until the latency period associated with the current instruction expires. When the execution unit becomes free, the next instruction may then be scheduled for execution on it. As for data dependencies, it is known that instructions are often

dependent on results generated by previous instructions. These results are stored within the reservation station. See page 254 (execute step) and page 256. When the operands are stored for a particular instruction, that instruction may be executed. Until the operands are ready, the instruction may not be executed. Therefore, dependencies are taken into consideration when scheduling.

d) a checker coupled to the execution unit to determine whether each instruction of the plurality of instructions has executed successfully, and coupled to the replay queue to dispatch to the replay queue each instruction that has not executed successfully. See Fig.7 and column 9, lines 50-53.

e) Sager has not taught a counter to count a number of times an instruction has one of executed and replayed, wherein independent instructions and associated dependent instructions are executed if the counter is less than a predetermined value and if the counter exceeds the predetermined value the instruction is prevented from executing until data required by the instruction is available. However, Heath has taught the general idea of when an instruction is erroneously executed, the instruction will be re-executed a predetermined number of times. When that number is exceeded, corrective measures have to be taken. See column 2, lines 1-14. Heath has not taught what the corrective measures include. However, Hennessy has taught that it is proper to stall the execution of an instruction until data required by the instruction is available. See page 152-154 of Hennessy. A person of ordinary skill in the art would have recognized that by using the idea of Heath, a counter could be implemented in Sager so that a predetermined number of replays are attempted, thereby trying to solve the execution error through re-execution. However, when the predetermined number is exceeded, the re-execution

of the erroneous instruction would be suspended so that other instruction which would possibly execute correctly would use the system resources instead of the continuously erroneous instruction. In addition, by using the idea of Hennessy, execution of the erroneous instruction would be delayed until its required data is available, at which point it may be re-executed again. This would increase efficiency of the system in that a lot of time and resources are not wasted trying to repeatedly execute an error-causing instruction. If the error is not resolved after a certain amount of time, then its re-execution should be suspended and corrective measures should be taken (waiting for its data to be available). This way, other instructions may be executed during the time which would have previously been spent executing the erroneous instruction. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Sager to include a counter as taught by Heath which allows for instruction replay a predetermined number of times before requiring corrective measures and to have those corrective measures be stalling the execution of the instruction until its data is available, as taught by Hennessy, so that other instructions may execute instead. Furthermore, from column 9, lines 31-36, column 10, lines 47-52, and column 12, lines 4-8, Sager has made it clear that when an independent instruction needs to be replayed, its dependent instructions also need to be replayed.

13. Referring to claim 2, Sager in view of Heath and further in view of Hennessy has taught a processor as described in claim 1. Sager has further taught an allocator/renamer coupled to the replay queue to allocate and rename those of a plurality of resources needed by the instruction. See the renamer component in Fig.7 and column 8, lines 33-50.

14. Referring to claim 3, Sager in view of Heath and further in view of Hennessy has taught a

processor as described in claim 2. Sager has further taught a front end coupled to the allocator/renamer to provide the plurality of instructions to the allocator/renamer. See Fig.7 and column 8, lines 29-32.

15. Referring to claim 4, Sager in view of Heath and further in view of Hennessy has taught a processor as described in claim 2. Sager has further taught a retire unit to retire, the plurality of instructions, coupled to the checker to receive those of the plurality of instructions that have executed successfully, and coupled to the allocator/renamer to communicate a de-allocate signal to the allocator/renamer. See Fig.7, Fig.8, and column 14, lines 12-20. Also, it is inherent that when an instruction is retired, its resources (i.e. registers) are deallocated so that another instruction has the option to use them (as opposed to resources continuing to be allocated to an instruction which no longer requires them because that instruction has completed). If these resources were not deallocated, then they would not be available to the processor, thereby inhibiting execution.

16. Referring to claim 6, Sager in view of Heath and further in view of Hennessy has taught a processor as described in claim 1. Sager has further taught:

a) at least one cache system on a die of the processor. See the instruction cache and data cache in Fig.7. Also, see Fig.3 and column 7, lines 25-26.

b) a plurality of external memory devices. See Fig.2 and note the use of external main memory and hard disk storage.

c) a memory request controller coupled to the execution unit to obtain a plurality of data from the at least one cache system and the plurality of external memory devices and to provide the plurality of data to the execution unit. See Fig.6 and note that data can be provided from the data

cache 310 to the ALU functional unit 300. It is inherent that if data is to be supplied to the execution units, then it must first be retrieved.

17. Referring to claim 7, Sager in view of Heath and further in view of Hennessy has taught a processor as described in claim 6. Sager has further taught that the at least one cache system comprises a first level cache system and a second level cache system. See Fig.2.

18. Referring to claim 8, Sager in view of Heath and further in view of Hennessy has taught a processor as described in claim 6. Sager has further taught that the external memory devices comprise at least one of a third level cache system, a main memory, and a disk memory. See Fig.2.

19. Referring to claim 9, Sager in view of Heath and further in view of Hennessy has taught a processor as described in claim 1. Sager has further taught a staging queue coupled between the checker and the scheduler. See the delay component in Fig.7. Also, see column 36-46, and note that the delay element acts as a queue in that it holds a copy of an instruction for multiple clock cycles until the same instruction completes all stages of execution.

20. Referring to claim 10, Sager in view of Heath and further in view of Hennessy has taught a processor as described in claim 1. Sager has further taught that the checker comprises a scoreboard to maintain a status of a plurality of resources. See Fig.8 and column 13, lines 18-31.

21. Referring to claim 17, Sager has taught a method comprising:

a) receiving an instruction of a plurality of instructions. See Fig.7 and column 9, lines 36-43 and note that the checker receives instructions from a delay unit.

b) placing the instruction in a queue with other instructions of the plurality of instructions. See Fig.7 and column 9, lines 50-53, and note that the checker places instructions in a queue (buffer).

Recall that this buffer can hold multiple instructions. See column 9, line 63, to column 10, line 2.

c) speculatively re-ordering those of the plurality of instructions in a scheduler. See column 8, lines 52-63. Sager has not explicitly taught that the re-ordering is done in a scheduler based on data dependencies and instruction latencies. However, Sager has taught that the scheduler may include a reservation station. See column 8, lines 52-63. As is known in the art, and as supported by Hennessy, a reservation station schedules instructions based on latencies and dependencies. For example, regarding latencies, see page 255-256 of Hennessy, and note that the reservation station tracks which execution units are busy. Clearly, if a unit is busy, it will not be able to execute a next instruction until the latency period associated with the current instruction expires. When the execution unit becomes free, the next instruction may then be scheduled for execution on it. As for data dependencies, it is known that instructions are often dependent on results from previous instructions. These results are stored within the reservation station. See page 254 (execute step) and page 256. When the operands are stored for a particular instruction, that instruction may be executed. Until the operands are ready, the instruction may not be executed. Therefore, dependencies are taken into consideration when scheduling (and note that instructions may not execute in order (see page 258 of Hennessy).

d) dispatching one of the plurality of instructions to an execution unit to be executed. See column 8, lines 64-67.

e) executing the instruction. See column 8, lines 64-67.

f) determining whether the instruction executed successfully. See Fig.7 and column 9, lines 31-36, and note the checker verifies the instruction's execution.

g) routing the instruction and all associated dependent instructions back to the queue if the instruction did not execute successfully. See column 9, lines 50-53. Also, see column 9, lines 31-36, column 10, lines 47-52, and column 12, lines 4-8. From these passages, Sager has made it clear that when an independent instruction needs to be replayed, its dependent instructions also need to be replayed.

h) retiring the instruction if the instruction executed successfully and allowing the instruction's associated dependent instructions to execute. See Fig.7, Fig.8, and column 14, lines 12-20. In addition, see column 10, lines 52-56, and column 12, lines 32-37. More specifically, if an independent instruction has executed unsuccessfully, then the data it has produced will be incorrect, causing itself and all dependent instructions to be replayed. However, if the independent instruction executes successfully, then it will have produced correct data and the dependent instructions will no longer need to be replayed, i.e., they can execute successfully (as long as they themselves are not incorrectly processed for some reason).

i) Sager has not taught a counting a number of times an instruction has one of executed and replayed, wherein the instruction and associated dependent instructions are executed if the number of times the instruction has one of executed or replayed is less than a predetermined value and if the number of times the instruction has one of executed and replayed exceeds the predetermined value the instruction is prevented from executing until data required by the instruction is available. However, Heath has taught the general idea of when an instruction is erroneously executed, the instruction will be re-executed a predetermined number of times. When that number is exceeded, corrective measures have to be taken. See column 2, lines 1-14. Heath has not taught what the corrective measures include. However, Hennessy has taught that

it is proper to stall the execution of an instruction until data required by the instruction is available. See page 152-154 of Hennessy. A person of ordinary skill in the art would have recognized that by using the idea of Heath, a counter could be implemented in Sager so that a predetermined number of replays are attempted, thereby trying to solve the execution error through re-execution. However, when the predetermined number is exceeded, the re-execution of the erroneous instruction would be suspended so that other instruction which would possibly execute correctly could use the system resources instead of the continuously erroneous instruction. In addition, by using the idea of Hennessy, execution of the erroneous instruction would be delayed until its required data is available, at which point it may be re-executed again. This would increase efficiency of the system in that a lot of time is not wasted trying to repeatedly execute an error-causing instruction. If the error is not resolved after a certain amount of time, then its re-execution should be suspended and corrective measures should be taken (waiting for its data to be available). This way, other instructions may be executed during the time which would have previously been spent executing the erroneous instruction. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Sager to include a counter as taught by Heath which allows for instruction replay a predetermined number of times before requiring corrective measures and to have those corrective measures be stalling the execution of the instruction until its data is available, as taught by Hennessy, so that other instructions may execute instead. Furthermore, from column 9, lines 31-36, column 10, lines 47-52, and column 12, lines 4-8, Sager has made it clear that when an independent instruction needs to be replayed, its dependent instructions also need to be replayed.

22. Referring to claim 18, Sager in view of Heath and further in view of Hennessy has taught

a method as described in claim 17. Sager has further taught allocating those of a plurality of system resources needed by the instruction. See column 13, lines 61-64.

23. Claims 11-15 are rejected under 35 U.S.C. 103(a) as being unpatentable over Sager in view of Heath in view of Hennessy, as applied above, and further in view of Johnson, Superscalar Microprocessor Design, 1990, (herein referred to as Johnson). In addition Hennessy is used to show a teaching of how reservation stations are used in scheduling instructions for execution.

24. Referring to claim 11, Sager has taught a processor comprising:

a) a replay queue to receive a plurality of instructions. See the buffer in Fig.7 and column 9, line 50, to column 10, line 2.

b) at least two execution units to execute the plurality of instructions. See Fig.7 and Fig.5, and note the multiple execution cores.

c) Sager has taught a first scheduler (see Fig.7 and column 10, lines 7-32, and note that the combination of the scheduler and mux selects one of multiple instructions to send to the execution core. It can be seen that the output instruction of the buffer (replay queue) goes to the mux portion of the scheduler, wherein the mux will eventually send that instruction to the execution unit). Sager has not taught **at least two schedulers** coupled between the replay queue and the execution units to schedule instructions for execution based on data dependencies and instruction latencies. However, Sager has taught that the scheduler may include a reservation station. See column 8, lines 52-63. As is known in the art, and as supported by Hennessy, a reservation station schedules instructions based on latencies and dependencies. For example,

regarding latencies, see page 255-256 of Hennessy, and note that the reservation station tracks which execution units are busy. Clearly, if a unit is busy, it will not be able to execute a next instruction until the latency period associated with the current instruction expires. When the execution unit becomes free, the next instruction may then be scheduled for execution on it. As for data dependencies, it is known that instructions are often dependent on results from previous instructions. These results are stored within the reservation station. See page 254 (execute step) and page 256. When the operands are stored for a particular instruction, that instruction may be executed. Until the operands are ready, the instruction may not be executed. Therefore, dependencies are taken into consideration when scheduling. In addition, Hennessy has shown that multiple reservation stations (at least two) may be used to schedule instructions, each used for a different type of instruction (floating-point, integer). Johnson has taught this is beneficial because the logic associated with multiple reservation stations would be simpler than with one large scheduler (reservation station) for multiple reasons. See page 134, lines 1-22 of Johnson. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Sager in view Heath in view of Hennessy such that Sager includes at least two schedulers, thereby simplifying the logic associated with the schedulers.

d) a checker coupled to the execution unit to determine whether each instruction of the plurality of instructions has executed successfully, and coupled to the replay queue to dispatch to the replay queue each instruction that has not executed successfully. See Fig.7 and column 9, lines 50-53.

e) Sager has not taught a counter to count a number of times an instruction has one of executed and replayed, wherein independent instructions and associated dependent instructions are

executed if the counter is less than a predetermined value and if the counter exceeds the predetermined value the instruction is prevented from executing until data required by the instruction is available. However, Heath has taught the general idea of when an instruction is erroneously executed, the instruction will be re-executed a predetermined number of times. When that number is exceeded, corrective measures have to be taken. See column 2, lines 1-14. Heath has not taught what the corrective measures include. However, Hennessy has taught that it is proper to stall the execution of an instruction until data required by the instruction is available. See page 152-154 of Hennessy. A person of ordinary skill in the art would have recognized that by using the idea of Heath, a counter could be implemented in Sager so that a predetermined number of replays are attempted, thereby trying to solve the execution error through re-execution. However, when the predetermined number is exceeded, the re-execution of the erroneous instruction would be suspended so that other instruction which would possibly execute correctly could use the system resources instead of the continuously erroneous instruction. In addition, by using the idea of Hennessy, execution of the erroneous instruction would be delayed until its required data is available, at which point it may be re-executed again. This would increase efficiency of the system in that a lot of time is not wasted trying to repeatedly execute an error-causing instruction. If the error is not resolved after a certain amount of time, then its re-execution should be suspended and corrective measures should be taken (waiting for its data to be available). This way, other instructions may be executed during the time which would have previously been spent executing the erroneous instruction. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Sager to include a counter as taught by Heath which allows for instruction replay a

predetermined number of times before requiring corrective measures and to have those corrective measures be stalling the execution of the instruction until its data is available, as taught by Hennessy, so that other instructions may execute instead. Furthermore, from column 9, lines 31-36, column 10, lines 47-52, and column 12, lines 4-8, Sager has made it clear that when an independent instruction needs to be replayed, its dependent instructions also need to be replayed.

25. Referring to claim 12, Sager in view of Heath in view of Hennessy and further in view of Johnson has taught a processor as described in claim 11. Sager has further taught a plurality of memory devices coupled to the execution units such that the checker determines whether the instruction has executed successfully based on a plurality of information provided by the memory devices. See Fig.8 and column 13, lines 18-31, and note that the scoreboard is a memory device that allows the checker to determine whether an instruction's execution was successful. The scoreboard's operation per cycle is dependent on the instruction that has been executed, which has been provided by the instruction cache (Fig.7) and accesses a register file (Fig.6).

26. Referring to claim 13, Sager in view of Heath in view of Hennessy and further in view of Johnson has taught a processor as described in claim 12. Sager has further taught an allocator/renamer coupled to the replay queue to allocate and rename those of a plurality of resources needed by the plurality of instructions. See the renamer component in Fig.7 and column 8, lines 33-50.

27. Referring to claim 14, Sager in view of Heath in view of Hennessy and further in view of Johnson has taught a processor as described in claim 13. Sager has further taught a front end coupled to the allocator/renamer to provide the plurality of instructions to the allocator/renamer.

See Fig.7 and column 8, lines 29-32.

28. Referring to claim 15, Sager in view of Heath in view of Hennessy and further in view of Johnson has taught a processor as described in claim 13. Sager has further taught a retire unit to retire the plurality of instructions, coupled to the checker to receive those of the plurality of instructions that have executed successfully, and coupled to the allocator/renamer to communicate a de-allocate signal to the allocator/renamer. See Fig.7, Fig.8, and column 14, lines 12-20. Also, it is inherent that when an instruction is retired, its resources (i.e. registers) are deallocated so that another instruction has the option to use them (as opposed to resources continuing to be allocated to an instruction which no longer requires them because that instruction has completed). If these resources were not deallocated, then they would not be available to the processor, thereby inhibiting execution.

29. Claims 1-4, 6, 9-10, and 17-19 are rejected under 35 U.S.C. 103(a) as being unpatentable over Merchant et al., U.S. Patent No. 6,212,626 (as disclosed in the previous Office Action and herein referred to as Merchant) in view of Heath, as applied above, and further in view of Hennessy, as applied above.

30. Referring to claim 1, Merchant has taught a processor comprising:

a) a replay queue to receive a plurality of instructions. See the replay system 70 in Fig.1. More specifically, stages 84 and 85 would make up the replay queue.

b) an execution unit to execute the plurality of instructions. See Fig.1, component 58.

c) a scheduler coupled between the replay queue and the execution unit to speculatively schedule instructions for execution based on data dependencies. See column 2, lines 39-46, and Fig.1, and

note that the combination of the scoreboard 54, scheduler 30, and mux 56 selects one of multiple instructions to send to the execution core based partly on if the sources are ready (data dependency checking). It should be seen that the output instruction of the replay queue goes to the mux, wherein the mux will eventually send that instruction to the execution unit. In addition, it should be realized that instructions are scheduled based on latencies. For instance, if a given functional unit is currently busy executing another instruction, then an instruction wanting to use that same unit must wait (i.e., it cannot be scheduled until the previous instruction is finished). Also, see column 4, lines 27-31, and note that latencies are involved in the scheduling process.

d) a checker coupled to the execution unit to determine whether each instruction of the plurality of instructions has executed successfully, and coupled to the replay queue to dispatch to the replay queue each instruction that has not executed successfully. See Fig.1, component 72.

e) Merchant has not taught a counter to count a number of times an instruction has one of executed and replayed, wherein independent instructions and associated dependent instructions are executed if the counter is less than a predetermined value and if the counter exceeds the predetermined value the instruction is prevented from executing until data required by the instruction is available. However, Heath has taught the general idea of when an instruction is erroneously executed, the instruction will be re-executed a predetermined number of times. When that number is exceeded, corrective measures have to be taken. See column 2, lines 1-14. Heath has not taught what the corrective measures include. However, Hennessy has taught that it is proper to stall the execution of an instruction until data required by the instruction is available. See page 152-154 of Hennessy. A person of ordinary skill in the art would have recognized that by using the idea of Heath, a counter could be implemented in Merchant so that a

predetermined number of replays are attempted, thereby trying to solve the execution error through re-execution. However, when the predetermined number is exceeded, the re-execution of the erroneous instruction would be suspended so that other instruction which would possibly execute correctly could use the system resources instead of the continuously erroneous instruction. In addition, by using the idea of Hennessy, execution of the erroneous instruction would be delayed until its required data is available, at which point it may be re-executed again. This would increase efficiency of the system in that a lot of time is not wasted trying to repeatedly execute an error-causing instruction. If the error is not resolved after a certain amount of time, then its re-execution should be suspended and corrective measures should be taken (waiting for its data to be available). This way, other instructions may be executed during the time which would have previously been spent executing the erroneous instruction. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Merchant to include a counter as taught by Heath which allows for instruction replay a predetermined number of times before requiring corrective measures and to have those corrective measures be stalling the execution of the instruction until its data is available, as taught by Hennessy, so that other instructions may execute instead. Furthermore, Merchant has made it clear that when an independent instruction needs to be replayed, its dependent instructions also need to be replayed. See Fig.6A-6E and column 6, line 46, to column 7, line 10. More specifically, note that the instruction written in cycle 2 (Fig.6B) is dependent on the instruction written in cycle 1 (Fig.6A). This is denoted by the use of the letter "D" in Fig.6B. Likewise, the instruction written in cycle 3 (Fig.6C) is dependent on the instructions written in cycles 1 and 2 (Fig.6A-B). This is denoted by the use of two letter "D's" in Fig.6C. Also, in Fig.6C, it is

determined that instruction 1 must be replayed (since it failed the check). As a result dependent instruction 2 and 3 must also be replayed, as shown in Fig.6D-E.

31. Referring to claim 2, Merchant in view of Heath and further in view of Hennessy has taught a processor as described in claim 1. Merchant has further taught an allocator/renamer coupled to the replay queue to allocate and rename those of a plurality of resources needed by the instruction. See Fig.1 and Fig.2 and note that the scoreboard deals with renaming and allocation.

32. Referring to claim 3, Merchant in view of Heath and further in view of Hennessy has taught a processor as described in claim 2. Merchant has further taught a front end coupled to the allocator/renamer to provide the plurality of instructions to the allocator/renamer. See Fig.1, component 52.

33. Referring to claim 4, Merchant in view of Heath and further in view of Hennessy has taught a processor as described in claim 2. Merchant has further taught a retire unit to retire, the plurality of instructions, coupled to the checker to receive those of the plurality of instructions that have executed successfully, and coupled to the allocator/renamer to communicate a de-allocate signal to the allocator/renamer. See Fig.1, component 62. Also, when an instruction is retired, its resources (i.e. registers) are deallocated so that another instruction has the option to use them. See column 3, lines 27-29.

34. Referring to claim 6, Merchant in view of Heath and further in view of Hennessy has taught a processor as described in claim 1. Merchant has further taught:

a) at least one cache system on a die of the processor. Note the disclosure of a cache in column 3, lines 53-54. For a cache miss to occur, a cache system must exist.

b) a plurality of external memory devices. See column 2, lines 24-27. The existence of main memory and/or hard disk is inherent. These types of slower, but larger memories are used to hold programs and data for execution.

c) a memory request controller coupled to the execution unit to obtain a plurality of data from the at least one cache system and the plurality of external memory devices and to provide the plurality of data to the execution unit. See Fig.1, bus 98, and column 2, lines 24-27. Note that a controller would be required to retrieve and send data along bus 98 to some memory device.

35. Referring to claim 9, Merchant in view of Heath and further in view of Hennessy has taught a processor as described in claim 1. Merchant has further taught a staging queue coupled between the checker and the scheduler. See Fig.1, components 80, 81, 82, and 83.

36. Referring to claim 10, Merchant in view of Heath and further in view of Hennessy has taught a processor as described in claim 1. Merchant has further taught that the checker comprises a scoreboard to maintain a status of a plurality of resources. See Fig.1, component 54, and Fig.2.

37. Referring to claim 17, Merchant has taught a method comprising:

a) receiving an instruction of a plurality of instructions. See Fig.1 and note that the checker receives an instruction via staging queue 80-83.

b) placing the instruction in a queue with other instructions of the plurality of instructions. See Fig.1 and note that the instruction may be placed in replay queue 84-85.

c) speculatively re-ordering those of the plurality of instructions in a scheduler based on data dependencies and instruction latencies. See column 2, lines 15-17, and lines 38-53. Note that instructions cannot be executed until their resources are available (data dependencies) and the

availability of these resources is dependent on the latencies of the instructions producing those resources. In addition, it should be realized that instructions are scheduled based on latencies. For instance, if a given functional unit is currently busy executing another instruction, then an instruction wanting to use that same unit must wait (i.e., it cannot be scheduled until the previous instruction is finished). Also, see column 4, lines 27-31, and note that latencies are involved in the scheduling process.

d) dispatching one of the plurality of instructions to an execution unit to be executed. See Fig.1 and column 2, lines 62-65.

e) executing the instruction. See Fig.1 and column 2, lines 62-66.

f) determining whether the instruction executed successfully. See column 3, lines 46-48.

g) routing the instruction and all associated dependent instructions back to the queue if the instruction did not execute successfully. See Fig.1 and column 3, lines 17-32. Also, see Fig.6A-6E and column 6, line 46, to column 7, line 10. More specifically, note that the instruction written in cycle 2 (Fig.6B) is dependent on the instruction written in cycle 1 (Fig.6A). This is denoted by the use of the letter "D" in Fig.6B. Likewise, the instruction written in cycle 3 (Fig.6C) is dependent on the instructions written in cycles 1 and 2 (Fig.6A-B). This is denoted by the use of two letter "D's" in Fig.6C. Also, in Fig.6C, it is determined that instruction 1 must be replayed (since it failed the check). As a result dependent instruction 2 and 3 must also be replayed, as shown in Fig.6D-E. And, replayed instructions are stored in a queue.

h) retiring the instruction if the instruction executed successfully and allowing the instruction's associated dependent instructions to execute. See Fig.1, component 62 and column 3, lines 23-27. Also, see Fig.6F-H, and column 7, lines 11-42. Note that the instruction is declared replay

safe (i.e., it passed the check and has executed successfully). As a result, in Fig.6G-H, the instruction's dependents are also allowed to execute and are declared replay safe (replay is not needed).

i) Merchant has not taught a counting a number of times an instruction has one of executed and replayed, wherein the instruction and associated dependent instructions are executed if the number of times the instruction has one of executed or replayed is less than a predetermined value and if the number of times the instruction has one of executed and replayed exceeds the predetermined value the instruction is prevented from executing until data required by the instruction is available. However, Heath has taught the general idea of when an instruction is erroneously executed, the instruction will be re-executed a predetermined number of times. When that number is exceeded, corrective measures have to be taken. See column 2, lines 1-14. Heath has not taught what the corrective measures include. However, Hennessy has taught that it is proper to stall the execution of an instruction until data required by the instruction is available. See page 152-154 of Hennessy. A person of ordinary skill in the art would have recognized that by using the idea of Heath, a counter could be implemented in Merchant so that a predetermined number of replays are attempted, thereby trying to solve the execution error through re-execution. However, when the predetermined number is exceeded, the re-execution of the erroneous instruction would be suspended so that other instruction which would possibly execute correctly could use the system resources instead of the continuously erroneous instruction. In addition, by using the idea of Hennessy, execution of the erroneous instruction would be delayed until its required data is available, at which point it may be re-executed again. This would increase efficiency of the system in that a lot of time is not wasted trying to

repeatedly execute an error-causing instruction. If the error is not resolved after a certain amount of time, then its re-execution should be suspended and corrective measures should be taken (waiting for its data to be available). This way, other instructions may be executed during the time which would have previously been spent executing the erroneous instruction. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Merchant to include a counter as taught by Heath which allows for instruction replay a predetermined number of times before requiring corrective measures and to have those corrective measures be stalling the execution of the instruction until its data is available, as taught by Hennessy, so that other instructions may execute instead. Furthermore, Merchant has made it clear that when an independent instruction needs to be replayed, its dependent instructions also need to be replayed. See Fig.6A-6E and column 6, line 46, to column 7, line 10. More specifically, note that the instruction written in cycle 2 (Fig.6B) is dependent on the instruction written in cycle 1 (Fig.6A). This is denoted by the use of the letter "D" in Fig.6B. Likewise, the instruction written in cycle 3 (Fig.6C) is dependent on the instructions written in cycles 1 and 2 (Fig.6A-B). This is denoted by the use of two letter "D's" in Fig.6C. Also, in Fig.6C, it is determined that instruction 1 must be replayed (since it failed the check). As a result dependent instruction 2 and 3 must also be replayed, as shown in Fig.6D-E.

38. Referring to claim 18, Merchant in view of Heath and further in view of Hennessy has taught a method as described in claim 17. Merchant has further taught allocating those of a plurality of system resources needed by the instruction. See the scoreboard in Fig.1 and column 2, lines 43-44.

39. Referring to claim 19, Merchant in view of Heath and further in view of Hennessy has

taught a method as described in claim 18. Merchant has further taught that retiring comprises:

a) de-allocating those of the plurality of system resources used by the instruction being retired. See column 3, lines 17-29.

b) removing the instruction and a plurality of related data from the queue. If an instruction is eligible for retirement, then there is no need to keep it in the queue, since it won't need to execute again. Therefore, it is inherent that the instruction along with its data will be removed. Otherwise, if instructions were not removed, once the replay queue fills up due to its finite storage space, it will stay full and no additional instructions would be able to be stored for replay purposes.

40. Claims 11-15 are rejected under 35 U.S.C. 103(a) as being unpatentable over Merchant in view of Heath in view of Hennessy, as applied above, and further in view of Johnson as applied above.

41. Referring to claim 11, Merchant has taught a processor comprising:

a) a replay queue to receive a plurality of instructions. See replay system 70 in Fig. 1. More specifically stages 84 and 85 would make up the replay queue.

b) at least two execution units to execute the plurality of instructions. See Fig. 1, component 58, and column 2, lines 65-66.

c) Merchant has taught a first scheduler coupled between the replay queue and the execution unit to speculatively schedule instructions for execution based on data dependencies (see column 2, lines 39-46, and Fig. 1, and note that the combination of the scoreboard 54, scheduler 30, and mux 56 selects one of multiple instructions to send to the execution core based partly on if the

sources are ready (data dependency checking). It should be seen that the output instruction of the replay queue goes to the mux, wherein the mux will eventually send that instruction to the execution unit). Merchant has not taught **at least two schedulers** coupled between the replay queue and the execution units to schedule instructions for execution based on data dependencies and instruction latencies. However, Hennessy has taught multiple schedulers (reservation stations) which schedule instructions based on latencies and dependencies. For example, regarding latencies, see page 255-256 of Hennessy, and note that the reservation station tracks which execution units are busy. Clearly, if a unit is busy, it will not be able to execute a next instruction until the latency period associated with the current instruction expires. When the execution unit becomes free, the next instruction may then be scheduled for execution on it. As for data dependencies, it is known that instructions are often dependent on results from previous instructions. These results are stored within the reservation station. See page 254 (execute step) and page 256. When the operands are stored for a particular instruction, that instruction may be executed. Until the operands are ready, the instruction may not be executed. Therefore, dependencies are taken into consideration when scheduling. In addition, Hennessy has shown that multiple reservation stations (at least two) may be used to schedule instructions, each used for a different type of instruction (floating-point, integer). Johnson has taught this is beneficial because the logic associated with multiple reservation stations would be simpler than with one large scheduler (reservation station) for multiple reasons. See page 134, lines 1-22 of Johnson. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Merchant in view Heath in view of Hennessy such that Merchant includes at least two schedulers, thereby simplifying the logic associated with the schedulers.

d) a checker coupled to the execution units to determine whether each instruction has executed successfully, and coupled to the replay queue to communicate each instruction that has not executed successfully. See Fig.1, component 72.

e) Merchant has not taught a counter to count a number of times an instruction has one of executed and replayed, wherein independent instructions and associated dependent instructions are executed if the counter is less than a predetermined value and if the counter exceeds the predetermined value the instruction is prevented from executing until data required by the instruction is available. However, Heath has taught the general idea of when an instruction is erroneously executed, the instruction will be re-executed a predetermined number of times. When that number is exceeded, corrective measures have to be taken. See column 2, lines 1-14. Heath has not taught what the corrective measures include. However, Hennessy has taught that it is proper to stall the execution of an instruction until data required by the instruction is available. See page 152-154 of Hennessy. A person of ordinary skill in the art would have recognized that by using the idea of Heath, a counter could be implemented in Merchant so that a predetermined number of replays are attempted, thereby trying to solve the execution error through re-execution. However, when the predetermined number is exceeded, the re-execution of the erroneous instruction would be suspended so that other instruction which would possibly execute correctly could use the system resources instead of the continuously erroneous instruction. In addition, by using the idea of Hennessy, execution of the erroneous instruction would be delayed until its required data is available, at which point it may be re-executed again. This would increase efficiency of the system in that a lot of time is not wasted trying to repeatedly execute an error-causing instruction. If the error is not resolved after a certain amount

of time, then its re-execution should be suspended and corrective measures should be taken (waiting for its data to be available). This way, other instructions may be executed during the time which would have previously been spent executing the erroneous instruction. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Merchant to include a counter as taught by Heath which allows for instruction replay a predetermined number of times before requiring corrective measures and to have those corrective measures be stalling the execution of the instruction until its data is available, as taught by Hennessy, so that other instructions may execute instead. Furthermore, Merchant has made it clear that when an independent instruction needs to be replayed, its dependent instructions also need to be replayed. See Fig.6A-6E and column 6, line 46, to column 7, line 10. More specifically, note that the instruction written in cycle 2 (Fig.6B) is dependent on the instruction written in cycle 1 (Fig.6A). This is denoted by the use of the letter "D" in Fig.6B. Likewise, the instruction written in cycle 3 (Fig.6C) is dependent on the instructions written in cycles 1 and 2 (Fig.6A-B). This is denoted by the use of two letter "D's" in Fig.6C. Also, in Fig.6C, it is determined that instruction 1 must be replayed (since it failed the check). As a result dependent instruction 2 and 3 must also be replayed, as shown in Fig.6D-E.

42. Referring to claim 12, Merchant in view of Heath in view of Hennessy and further in view of Johnson has taught a processor as described in claim 11. Merchant has further taught a plurality of memory devices coupled to the execution units such that the checker determines whether the instruction has executed successfully based on a plurality of information provided by the memory devices. See Fig.2 and note that the scoreboard is a memory device that allows the checker to determine whether an instruction's execution was successful. The scoreboard's

operation per cycle is dependent on the instruction that has been executed, which has been ultimately provided by the instruction queue memory (Fig.1).

43. Referring to claim 13, Merchant in view of Heath in view of Hennessy and further in view of Johnson has taught a processor as described in claim 12. Merchant has further taught an allocator/renamer coupled to the replay queue to allocate and rename those of a plurality of resources needed by the plurality of instructions. See Fig.1 and Fig.2 and note that the scoreboard deals with renaming and allocation.

44. Referring to claim 14, Merchant in view of Heath in view of Hennessy and further in view of Johnson has taught a processor as described in claim 13. Merchant has further taught a front end coupled to the allocator/renamer to provide the plurality of instructions to the allocator/renamer. See Fig.1, component 52.

45. Referring to claim 15, Merchant in view of Heath in view of Hennessy and further in view of Johnson has taught a processor as described in claim 13. Merchant has further taught a retire unit to retire the plurality of instructions, coupled to the checker to receive those of the plurality of instructions that have executed successfully, and coupled to the allocator/renamer to communicate a de-allocate signal to the allocator/renamer. See Fig.1, component 62. Also, when an instruction is retired, its resources (i.e. registers) are deallocated so that another instruction has the option to use them. See column 3, lines 27-29.

46. Claims 5 and 19 are rejected under 35 U.S.C. 103(a) as being unpatentable over Sager in view of Heath in view of Hennessy, as applied above, and further in view of Baxter et al., U.S.

Patent No. 5,944,818 (as disclosed in the previous Office Action and herein referred to as Baxter).

47. Referring to claim 5, Sager in view of Heath and further in view of Hennessy has taught a processor as described in claim 4. Sager has not explicitly taught that the retire unit is further coupled to the replay queue to communicate a retire signal when one of the plurality of instructions is retired such that the retired instruction and a plurality of associated data are removed from the replay queue. However, Baxter has taught such a concept. See Fig.2 and column 3, lines 43-55, and note that upon retirement, the corresponding instruction entry in the replay queue (MIQ) is discarded (via deallocation signal shown in Fig.2) since there is no longer a need to maintain the instruction. Likewise, when an instruction retires in Sager, there would be no need to maintain that instruction in the replay queue. Doing so would consume resources for no beneficial reason. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Sager in view of Baxter so that upon retirement of an instruction, the retire unit communicates a signal to the replay queue in order to remove that instruction and its associated data (such as branch prediction information as disclosed in column 5, lines 64-66) from the queue.

48. Referring to claim 19, Sager in view of Heath and further in view of Hennessy has taught a method as described in claim 18.

a) Sager has further taught that retiring comprises de-allocating those of the plurality of system resources used by the instruction being retired. It is inherent that when an instruction is retired, its resources (i.e. registers) are deallocated so that another instruction has the option to use them (as opposed to resources continuing to be allocated to an instruction which no longer requires

them because that instruction has completed). If these resources were not deallocated, then they would not be available to the processor, thereby inhibiting execution.

b) Sager has not explicitly taught that retiring comprises removing the instruction and a plurality of related data from the queue. However, Baxter has taught such a concept. See Fig.2 and column 3, lines 43-55, and note that upon retirement, the corresponding instruction entry in the replay queue (MIQ) is discarded (via deallocation signal shown in Fig.2) since there is no longer a need to maintain the instruction. Likewise, when an instruction retires in Sager, there would be no need to maintain that instruction in the replay queue. Doing so would consume resources for no beneficial reason. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Sager in view of Baxter so that upon retirement of an instruction, the retire unit communicates a signal to the replay queue in order to remove that instruction and its associated data (such as branch prediction information as disclosed in column 5, lines 64-66) from the queue.

49. Claim 16 is rejected under 35 U.S.C. 103(a) as being unpatentable over Sager in view of Heath in view of Hennessy in view of Johnson, as applied above, and further in view of Baxter, as applied above.

50. Referring to claim 16, Sager in view of Heath in view of Hennessy and further in view of Johnson has taught a processor as described in claim 15. Sager has not explicitly taught that the retire unit is further coupled to the replay queue to communicate a retire signal when one of the plurality of instructions is retired such that the retired instruction and a plurality of associated data are removed from the replay queue. However, Baxter has taught such a concept. See Fig.2

and column 3, lines 43-55, and note that upon retirement, the corresponding instruction entry in the replay queue (MIQ) is discarded (via deallocation signal shown in Fig.2) since there is no longer a need to maintain the instruction. Likewise, when an instruction retires in Sager, there would be no need to maintain that instruction in the replay queue. Doing so would consume resources for no beneficial reason. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Sager in view of Baxter so that upon retirement of an instruction, the retire unit communicates a signal to the replay queue in order to remove that instruction and its associated data (such as branch prediction information as disclosed in column 5, lines 64-66) from the queue.

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to David J. Huisman whose telephone number is (703) 305-7811. The examiner can normally be reached on Monday-Friday (8:00-4:30).

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Eddie Chan can be reached on (703) 305-9712. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

DJH
David J. Huisman
July 9, 2004

Eddie Chan

EDDIE CHAN
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100